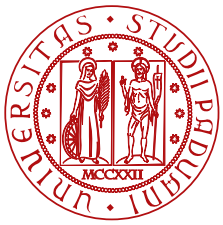


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

Simulazione del Controllo Digitale di un Convertitore Boost

Relatore
Prof. Simone Buso

Laureando
Brunelli Marco

ANNO ACCADEMICO 2023-2024

Data di laurea 25/09/2024

Indice

Capitolo 1

Specifiche di progetto

Questo progetto non ha stretti requisiti legati ad un progetto reale per un'azienda, ma si andranno comunque a definire delle specifiche realistiche anche se un po' più flessibili.

Questo convertitore alza la tensione da 50V in ingresso a 70V in uscita e alimenta un carico di 2.5Ω. La risposta al gradino deve essere il più possibile simile a una risposta del primo ordine con un tempo di assestamento di meno di 3 ms e non deve produrre overshoot troppo elevati. Il convertitore deve riuscire a regolare la tensione anche se il carico aumenta del 50% senza avere picchi di tensione maggiore del 25% rispetto al valore al riferimento di tensione. Si aggiunge anche una specifica per il ripple picco-picco della corrente sull'induttore che non deve essere maggiore del 30% del valore nominale di corrente. Il ripple di tensione in uscita deve essere minore del 5% della tensione nominale.

Capitolo 2

Convertitore Boost sincrono

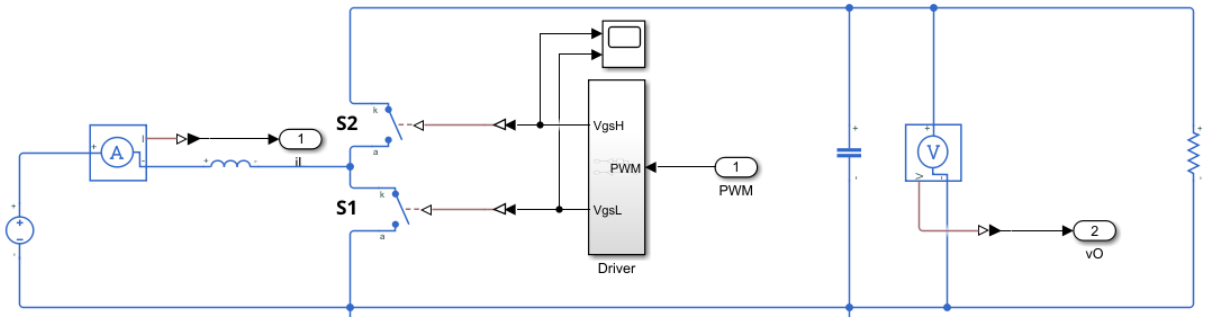


Figura 2.1: Boost in matlab

Un convertitore boost [1] è un tipo di convertitore DC-DC che serve per aumentare la tensione in uscita rispetto ad una più bassa tensione in ingresso. Il convertitore in questione è di tipo sincrono e il suo vantaggio principale è dato dalla minore dissipazione di potenza dell'interruttore MOSFET rispetto al diodo presente del boost asincrono. Grazie a questo, non c'è bisogno di aggiungere sistemi di raffreddamento complessi e si può ottenere una maggiore efficienza. Il diodo ha una caduta di tensione intorno ai 0.6 V mentre la V_{ds} di un mosfet, è molto bassa, quasi approssimabile a 0 V. Lo svantaggio maggiore è dovuto al bisogno di un driver che eviti cortocircuiti causati dalla chiusura di entrambi gli interruttori elettronici e fornisca abbastanza corrente per una commutazione veloce. Si utilizzerà il convertitore solamente in modo continuo, quindi la corrente sull'induttore non sarà mai negativa o pari a 0. Per capire il suo funzionamento in modalità continua si divide il circuito nelle due topologie.

- Quando l'interruttore S1 è chiuso e S2 è aperto l'induttore si carica grazie alla tensione di ingresso mentre il condensatore si scarica sul carico.
- Quando l'interruttore S1 è aperto e S2 è chiuso la corrente proveniente dalla sorgente e dall'induttore scorre verso il carico e il condensatore viene caricato.

Si inizia a trovare i dati per calcolare il dimensionamento dell'induttore in funzione del ripple voluto. Si calcola il massimo ripple accettato per la massima corrente in uscita; questo avverrà quando il carico sarà minimo quindi pari a 1.25Ω

$$I_{out-max} = \frac{V_{out}}{R_{out-min}} = \frac{70V}{1.25\Omega} = 56A$$

$$\Delta I_L = 0.30 \cdot I_{out-max} = 0.30 \cdot 56A = 16.8A$$

Questo significa che la corrente attraverso l'induttore dovrà fluttuare al massimo di ±8.4 A

Il duty cycle viene calcolato per un convertitore con un efficienza del 100% con la formula:

$$D = 1 - \frac{V_{IN}}{V_{out}} = 1 - \frac{50V}{70V} = 0.2857$$

Si procede con i dati trovati al calcolo dell'induttanza

$$L = \frac{V_{IN} \cdot D}{f_s \cdot \Delta I_L} = \frac{50V \cdot 0.2857}{25000Hz \cdot 16.8A} = 34\mu H$$

Nella formula è stata utilizzata la frequenza di switching dell'onda PWM generata dal µcontrollore, pari a $25kHz$

In uscita si avrà una componente continua con sommate le armoniche ai multipli delle frequenza di switching del MOSFET. C'è bisogno di un filtro LC che grazie alla sua attenuazione di 40dB/dec consente di tagliare efficacemente le armoniche. Si deve sopratutto attenuare la prima armonica alla frequenza di switching, perché ha una ampiezza maggiore rispetto alla componente continua.

Per ottenere il filtro si aggiunge un condensatore in parallelo all'uscita e si utilizza come induttore quello utilizzato per alzare la tensione. Si è già calcolato il valore dell'induttanza; ora si trova il valore del condensatore minimo di uscita:

$$C_{OUTmin} = \frac{I_{OUTmax} \cdot D}{f_s \cdot \Delta V_{OUT}} = \frac{56A \cdot 0.2857}{25000Hz \cdot 3.5V} = 182.8\mu F$$

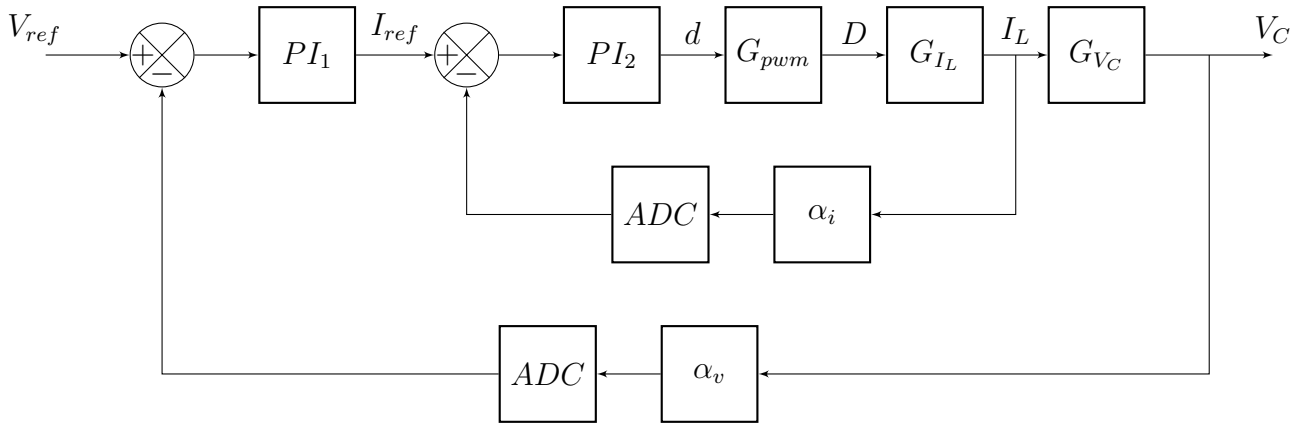
Per ottenere questa capacità si utilizza un grosso condensatore elettrolitico con in parallelo vari condensatori ceramici. Il condensatore elettrolitico grazie alla sua grande capacità riesce a stabilizzare le variazioni lente della tensione di uscita mentre i condensatori ceramici con la loro bassa ESR riescono a filtrare le alte frequenze.

Lo svantaggio principale del del filtro LC è la presenza di un picco in frequenza che si trova alla frequenza di risonanza pari a:

$$f_{\mathrm{r}} = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{34 \cdot 10^{-6} \cdot 182.8 \cdot 10^{-6}}} = 2018Hz$$

La frequenza di risonanza del filtro non deve essere troppo bassa per non interferire con il controllo di corrente. Idealmente dovrebbe stare al di sopra della frequenza di taglio del controllo, e al di sotto della frequenza di modulazione per offrire l'attenuazione desiderata. Inoltre, nel boost, la frequenza dipende anche dal duty-cycle, per via del fatto che L e C non sono collegati tra loro: c'è di mezzo il convertitore. Quindi questo calcolo è molto approssimato.

2.1 Controllo a doppio anello PI



- PI_1 è il primo controllo PI che controlla la tensione
- PI_2 è il secondo controllo PI che controlla la corrente
- G_{pwm} rappresenta la funzione di trasferimento dal valore del registro della periferica PWM al valore del duty cycle reale.
- G_{I_L} rappresenta la funzione di trasferimento che descrive la relazione tra il duty cycle reale e la corrente dell'induttore (i_L). Questo blocco simula il comportamento dell'induttanza nel circuito e come la corrente varia in base al segnale PWM applicato.
- G_{V_C} rappresenta la funzione di trasferimento che descrive la relazione tra la corrente dell'induttore (I_L) e la tensione di uscita del condensatore (V_C) e simula il comportamento del condensatore
- ADC rappresentano il campionamento che avverrà sempre alla metà della parte altra del segnale D[k] in modo da ricavare il valor medio di I_L e V_C. Il campionamento ha una frequenza di $25kHz$ pari a quella del segnale D[k].

- α_v e α_i sono costanti che adattano tensione e corrente al valore massimo misurabile dall'ADC.

I blocchi G_{V_C} e G_{I_L} verranno simulati da µVision discretamente. Si riportano le equazioni alle differenze che verranno utilizzate:

$I_L[k] = I_L[k-1] + D[k](\frac{V_{IN}}{L}) \cdot T_s + (1 - D[k])(\frac{V_{IN} - V_C[k-1]}{L}) \cdot T_s$

$V_C[k] = V_C[k-1] + D[k] \cdot (\frac{-V_C[k-1]}{RC}) \cdot T_s + (1 - D[k])(\frac{I_L[k] \cdot \frac{-V_C[k-1]}{R}}{C}) \cdot T_s$

T_s è il tempo che intercorre tra i vari istanti simulati.

$D[k]$ rappresenta lo stato del segnale PWM e sarà pari a 1 quando il segnale è alto e pari a 0 quando è basso.

Per riuscire a controllare il convertitore boost è necessario avere 2 zeri perché sono presenti due variabili di stato date dalla corrente nell'induttore e la tensione sul condensatore di uscita. Si può utilizzare un singolo controllore PID, ma il controllo a doppio anello ha più vantaggi; consente di limitare la corrente in uscita così da evitare la distruzione dei componenti se è presente un cortocircuito in uscita. Consente anche di separare i tempi di risposta, la corrente ha una dinamica più veloce rispetto alla tensione; in questo modo si possono ottimizzare i guadagni di un singolo anello per massimizzare le prestazioni del controllo. L'anello esterno è utilizzato per la dinamica lenta quindi per il controllo della tensione in uscita, il suo compito è quello di fornire un riferimento per l'anello interno. L'anello interno è responsabile della dinamica veloce e controlla la corrente.

Capitolo 3

µControllore

Come processore viene utilizzato uno della famiglia STM32F103 che come core sfrutta un Cortex-M3, con una massima velocità di 72 MHz. Questi tipi di processori sono disponibili in molteplici package e con varie dimensioni di flash e ram. In particolare si sceglie il STM32F103C8 [2] [3] che è la variante più utilizzata perché ha un buon compromesso tra prestazioni e prezzo. Le periferiche principali presenti sono:

- 2 ADC SAR a 12 bit con fino a 16 canali
- un DMA con 7 canali
- 3 timer a 16 bit
- 64 Kbytes di memoria flash
- 20 Kbytes di SRAM

  Il segnale di clock dovrà essere fornito esternamente in quanto il segnale di clock generato internamente raggiunge una massima frequenza di 64MHz.

Si utilizza il software grafico CubeMX per inizializzare le periferiche e generare il codice base per il processore. Si possono fare delle modifiche direttamente al codice per evitare di dover sempre rieseguire CubeMX, ma bisogna ricordare di includere le librerie nella generazione del codice spuntando la casella apposita nel programma. Cliccando il tasto destro sul valore di una variabile usata per impostare una periferica si può andare sulla definizione di essa, questo porta alla libreria della periferica dove sono definiti i vari valori che la variabile può avere. Nell'header delle librerie sono commentate tutte le funzioni e variabili della libreria.

3.1 Timer

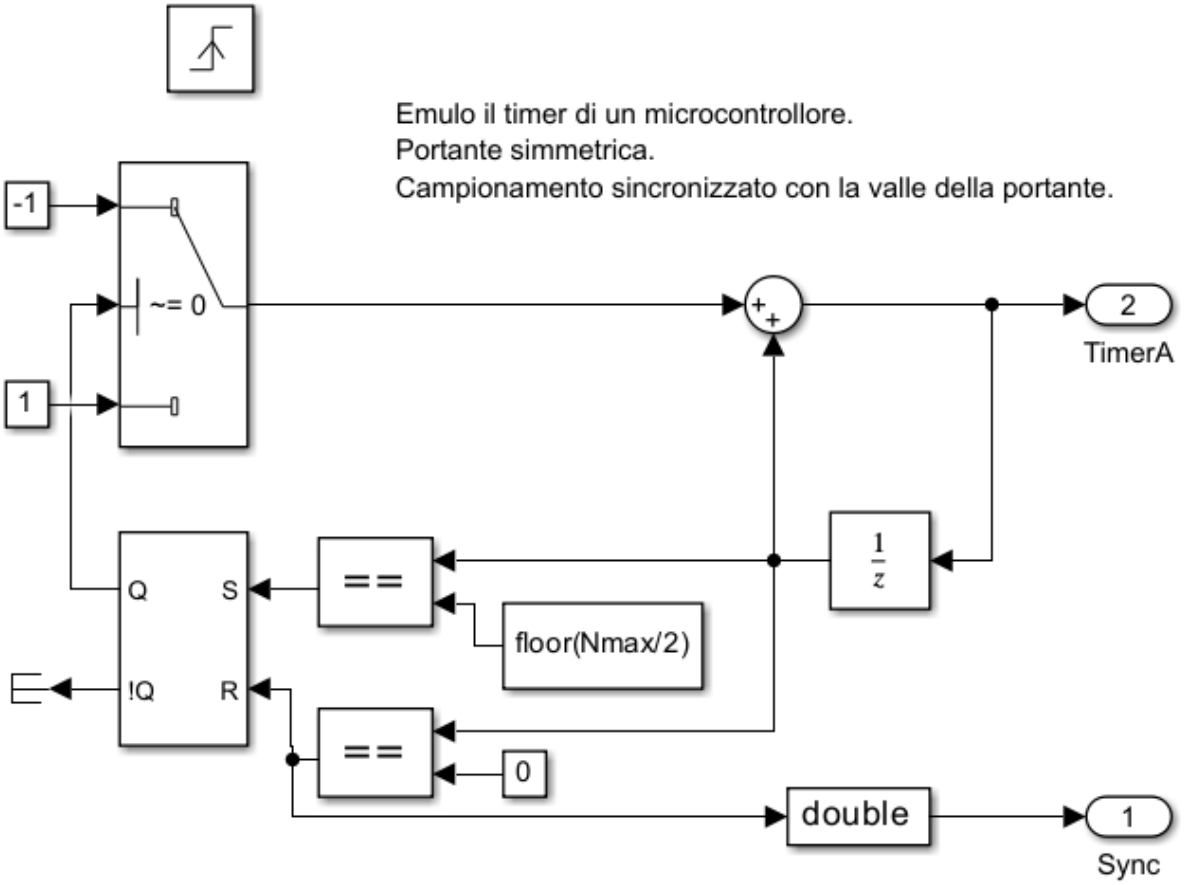


Figura 3.1: timer in matlab

Il timer del µcontrollore genera un'onda triangolare simmetrica a scalini. In ingresso al timer viene dato un segnale di clock pari a 72MHz che consente una risoluzione di 13.9 ns. Un clock maggiore porta ad una maggiore precisione del timer a scapito di un minore periodo del timer stesso. Questo avviene perché il valore massimo che può raggiungere il contatore del timer è di 65536. Il valore del timer viene aumentato di un unità ad ogni fronte positivo del segnale di clock finché non raggiunge il valore massimo impostato di 1440. Poi il valore del timer comincia a scendere finché non arriva a zero e tutto il processo ricomincia dall' inizio. Una volta raggiunto lo zero il timer manda anche un segnale di sync che attiva un interrupt dove l'ADC inizierà a campionare e si calcolerà il nuovo valore del duty cycle. Gli eventi generati dai timer possono essere collegati internamente al trigger di avvio dell'ADC, al trigger di iniezione e al trigger del DMA rispettivamente, questo consente di campionare precisamente nell'istante voluto.

3.2 PWM

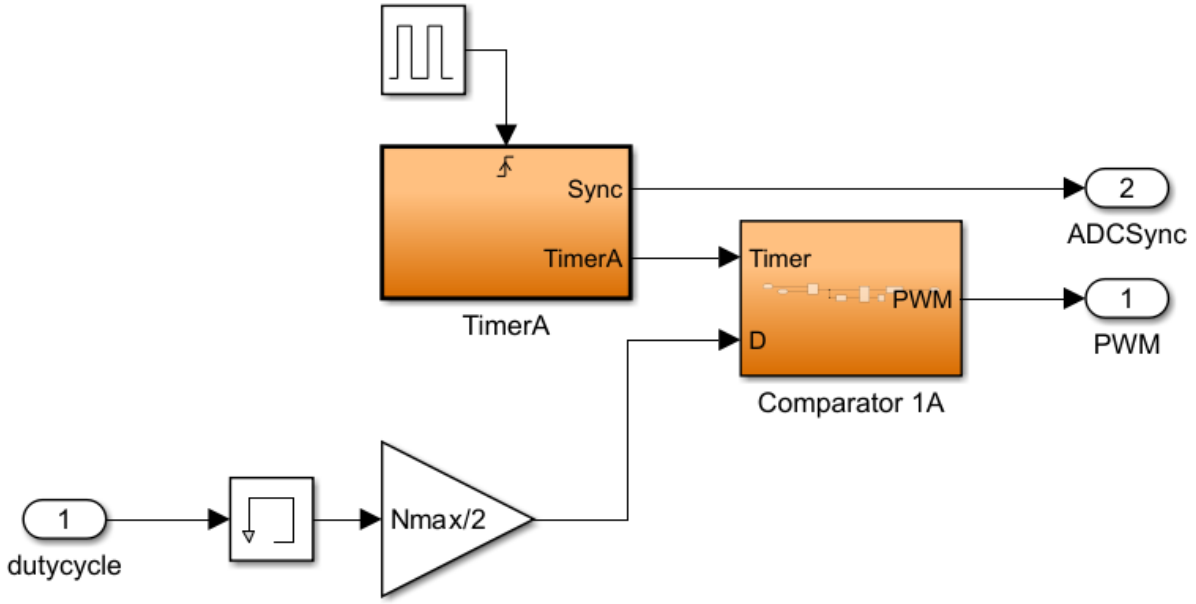


Figura 3.2: circuito PWM in matlab

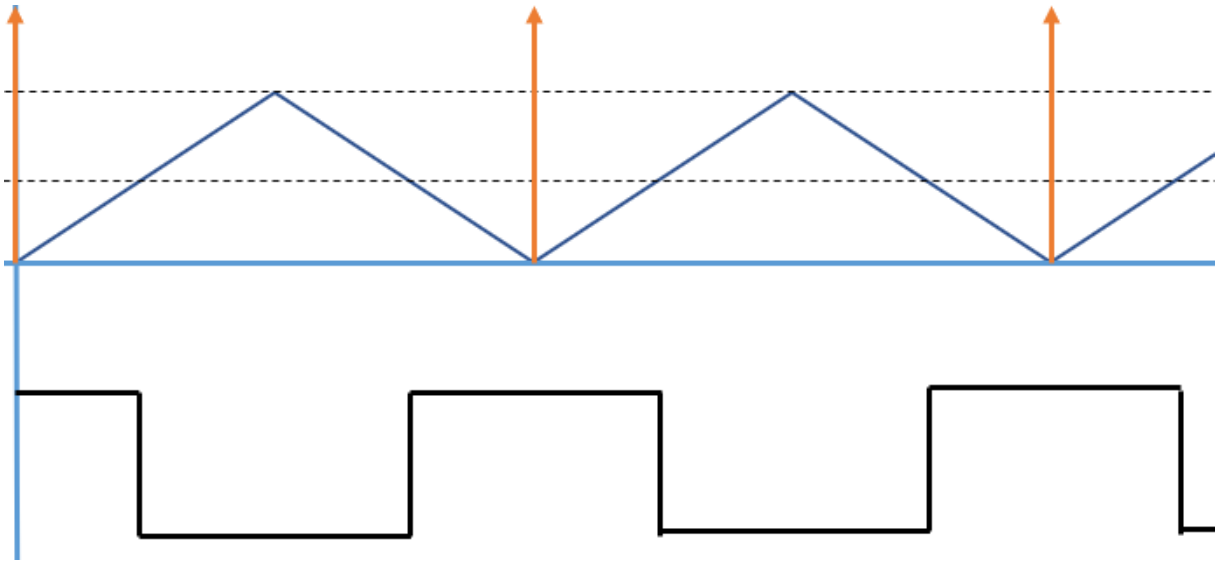


Figura 3.3: Il grafico sopra rappresenta il timer e la freccia arancione quando parte il trigger per l'ADC. Il grafico sotto rappresenta l'onda PWM

Un comparatore confronta il valore corrente del timer con un valore nel registro della periferica PWM. Quando il valore del registro è minore del valore del timer dalla periferica esce un segnale alto viceversa se il valore del registro è superiore o uguale a quello del timer, la periferica emette un segnale basso. Questo consente che l'onda quadra in uscita dal PWM sia perfettamente

simmetrica rispetto allo zero dell'onda triangolare del timer. Il periodo dell'onda sarà pari al periodo dell'onda triangolare portante del timer

$$PeriodoPWM = periodotimer * intervallotimer * 2 = 1440 * 13.9ns * 2 = 40\mu s$$

Un minore periodo del timer porta ad una frequenza di switching più elevata. Questa tecnica consente di campionare nella metà alta del duty cycle essendo che l'ADC campiona quando il timer raggiunge un valore pari a 0. Si può anche campionare a metà della parte bassa dell'onda PWM, semplicemente invertendo gli ingressi del comparatore. La tensione in uscita e la corrente avranno un certo ripple, campionando sempre a metà della parte alta del duty cycle si può estrapolare il valor medio della tensione/corrente

3.3 ADC

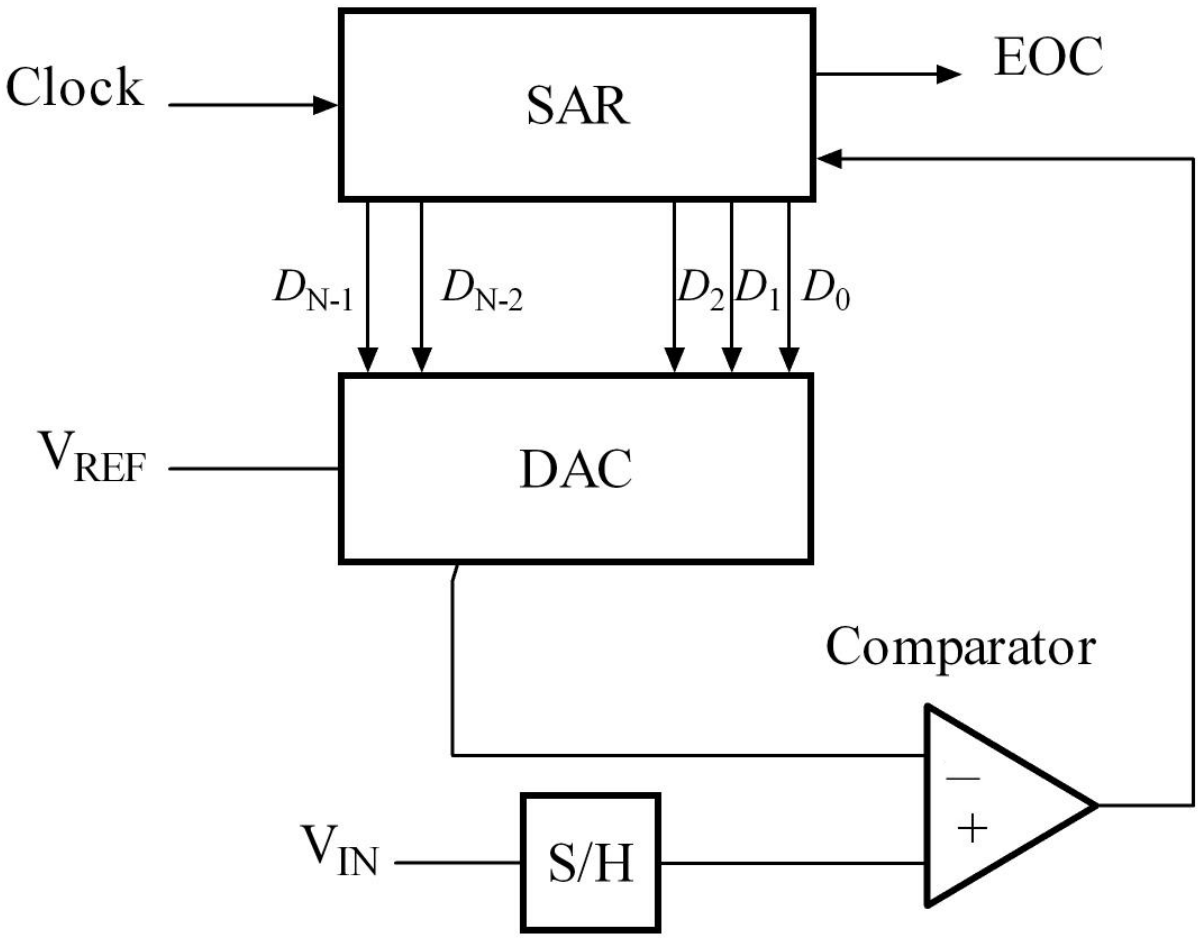


Figura 3.4: ADC SAR

Il µcontrollore possiede 2 Analog to Digital Converter ad approssimazioni successive a 12-bit con fino a 16 canali. Il campionamento di un singolo canale avviene con una velocità pari a 1.17 µs questo provocherà un lieve errore costante nel controllo. Un circuito sample and hold garantisce che la tensione in ingresso rimanga costante durante l'acquisizione. Il DAC inizialmente eroga una tensione pari a $V_{ref}/2$. Un comparatore confronta la tensione Vin con l'uscita del DAC e trasmette il risultato al registro ad approssimazioni successive se il segnale analogico è maggiore della Vin allora al bit verrà assegnato il valore 0, altrimenti il bit verrà lasciato ad 1. Questo processo continua per tutti gli altri 11 bit. L'ADC dovrà essere alimentata attraverso i PIN esterni V_{DDA} e V_{SSA} che saranno connessi ai pin di alimentazione del ucontrollore e avranno una tensione rispettivamente di 3.3V e 0V. Quindi la tensione massima che l'ADC può misurare è pari a 3.3V e la sua risoluzione è uguale a $V_{res} = \frac{3.3}{2^{12}} = 805.66\mu V$

3.4 DMA

L'approccio "Direct Memory Access" viene utilizzato per scrivere in memoria dei dati delle periferiche senza l'utilizzo della CPU, ma attraverso un controllore apposito chiamato DMAC. Il DMAC è una periferica hardware che si occupa autonomamente di gestire il trasferimento dati tra memoria e periferica controllando che la porzione di memoria non sia già utilizzata per consentire una scrittura senza causare conflitti e aspetta che il BUS sia libero prima di scrivere. Quando il trasferimento è completato il DMAC invia un interrupt alla CPU che eseguirà un codice che sfrutterà i dati salvati in memoria. In questo uprocessore il DMA, la CPU e le periferiche condividono lo stesso bus quindi ci sarà un trasferimento dalle periferiche al DMA e un trasferimento da DMA alla memoria. Altri µprocessori hanno un bus tra memoria e DMA e un altro bus tra periferiche e DMA questo consente di occupare per meno tempo il bus principale. Poiché il valore dei canali convertiti dall'ADC vengono memorizzati in un unico registro dati, è necessario utilizzare DMA per la conversione di più di un canale. Solo alla fine della conversione di un singolo canale viene generata una richiesta al DMA per effettuare il trasferimento dei dati convertiti dal registro ADC_DR alla posizione di destinazione selezionata dall'utente. Questo evita la perdita di dati nel registro ADC_DR.

Capitolo 4

Software

4.1 Keil-µVision 5

Il debugger di µVision [4] fornisce una simulazione completa del set di istruzioni per tutti i core Cortex-M0/M0+, Cortex-M3 e Cortex-M4. Tuttavia, è impossibile simulare il comportamento di tutte periferiche integrate per tutti i dispositivi. Solo in una piccola parte dei µcontrollori è possibile simulare completamente tutte le periferiche e sul sito di arm keil sono presenti tutte le periferiche simulabili per ogni µcontrollore [5]. Per attivare la simulazione del dispositivo bisogna cambiare i valori di DLL di dialogo e parametro secondo il dispositivo scelto all'interno nella finestra di debug [6]. Si andrà a scrivere in particolari registri virtuali associati a ciascuna periferica per interfacciarsi con la simulazione. Utilizzando il comando DIR VTREG è possibile controllare tutti i registi virtuali, con i loro relativi valori correnti, del µControllore simulato. Attraverso il menu a tendina Peripherals è possibile vedere, in una finestra grafica, tutte le periferiche simulate con i valori dei vari registri. Il codice usato per simulare usa una sintassi molto simile al linguaggio di programmazione ANSI C con molte funzioni specifiche.

4.2 Codice simulazione convertitore

Le funzioni di segnale ripetono operazioni in background mentre µVision esegue il programma target. Aiutano a simulare e testare I/O seriali e analogici, comunicazioni di porta e altri eventi esterni ripetitivi, come ingressi di segnale e impulsi. È obbligatorio chiamare la funzione integrata twatch almeno una volta per ritardare l'esecuzione e consentire a µVision di eseguire il programma target. Quando una funzione di segnale invoca la funzione twatch, entra nello stato di inattività per il numero di cicli di clock della CPU passati a twatch. Dopo che il programma utente ha eseguito il numero specificato di cicli di clock, la funzione di segnale torna in esecuzione. L'esecuzione riprende dall'istruzione successiva a twatch. Per il programma di simulazione

il twatch si comporta come un delay e consente di velocizzare la simulazione perché si esegue meno volte la funzione di segnale a scapito di una minore precisione di simulazione.

```
signal void LCR(float Vdc, float R) {

iL_old = 28;
uR_old = 70;

while (1) {

VA = (PORTA == 0x0100);          // fase alta dell'impulso

if (VA){DeltaI = (float)((Vdc)/(fcalc*L));}
else {DeltaI = (float)((Vdc - uR_old)/(fcalc*L));}

iL = iL_old + DeltaI;

if (VA){DeltaV = (float)((-uR_old/R)/(fcalc*C));}
else {DeltaV = (float)((iL-uR_old/R)/(fcalc*C));}
uR = uR_old + DeltaV;

ADC1_IN1 = (float)(FSRV*uR);
ADC1_IN0 = (float)(FSRI*iL);

iL_old = iL;
uR_old = uR;
twatch(6);

...
```

Si inizializzano la tensione sul condensatore e la corrente dell'induttore per evitare di dover simulare anche il transitorio per caricare i componenti. In base allo stato logico del segnale del duty cycle in ingresso al mosfet si capisce in che condizione topologica il circuito si trova. Essendo il clock di sistema pari a 72Mhz e twatch pari a 6 la funzione di simulazione verrà eseguita ogni 6 cicli di clock di sistema quindi si avrà una frequenza di calcolo pari a $fcalc = \frac{72Mhz}{6} = 12Mhz$. Per simulare il comportamento di induttanza e condensatore si esegue un integrale numerico con un un passo di integrazione pari a $\frac{1}{12Mhz} = 83ns$. Si calcola la variazione di corrente attraverso l'induttore utilizzando la legge di Faraday per cui $\Delta I_L = \frac{V_L}{L*fcalc}$

- Quando l'interruttore è chiuso l'induttore si porta alla tensione di ingresso e la corrente cresce con pendenza costante

- Quando l'interruttore è aperto l'induttore si porta alla tensione di ingresso meno la tensione sul condensatore e la corrente scende con pendenza negativa costante (dato che la tensione di uscita è maggiore di quella di ingresso)

Si aggiorna la corrente dell'induttore sommando il valore precedente della corrente e la variazione di corrente calcolata.

Per il condensatore si utilizza la legge del condensatore per cui $\Delta V_C = \frac{I_C}{C \cdot fcalc}$. Per la variazione di tensione del condensatore si trova la corrente passante con la legge della maglia

- Quando l'interruttore è chiuso il condensatore ha una corrente pari al negativo della corrente in uscita calcolata utilizzando il vecchio valore di tensione e la tensione scende

- Quando l'interruttore è aperto il condensatore ha una corrente pari alla corrente fornita dal l'induttore meno la corrente in uscita e la tensione sale

Alla fine si adatta la tensione e la corrente calcolate al fondo scala dell'ADC. Per simulare il funzionamento dell'amplificatore che andrà ad adattare i valori si moltiplicano i valori di corrente/tensione per un coefficiente calcolato come la frazione tra il valore massimo misurato dall'ADC e il valore massimo che si vuole misurare nel circuito. Per la corrente si è scelto un fondo scala di 100A mentre per la tensione un fondo scala di 100V per riuscire a misurare anche il ripple. É sempre meglio avere fondo scala più alto dei valori che ci si aspetta durante il controllo per evitare di superare il fondo scala in caso ci fossero problemi nel circuito causati da sovraccarichi, picchi imprevisti. I valori di tensione calcolati vengono salvati in dei registri virtuali appositi dei 3 canali dell'ADC.

```
...

a++;
if(a == 160000){
    ADC1_IN2 = (float)(1.4); //valore pari a 50V
    a=0;
}
if(a == 80000){
    ADC1_IN2 = (float)(1.925); //rifermento tensione
}
```

Si utilizza questo codice per cambiare il riferimento di tensione che si vuole ottenere in uscita. Il codice somma 1 alla variabile a ogni 6 cicli di clock quindi ogni 83ns. Ogni 80000 passi la tensione del riferimento cambia quindi si avrà un onda quadra con un periodo di $160000 \cdot 82ns = 13.28ms$. Si passa da un riferimento di 50V, la minima tensione che il convertitore può avere in uscita, a uno di 70V. È utile per studiare la risposta al gradino e tarare con facilità i guadagni del convertitore direttamente dal programma di debug senza fermare la simulazione. Si utilizza la stessa funzione per cambiare il carico.

```
FUNC void OnResetExec (void)  {
   LCR(50, 3);      //funzione di inizializzazione
}
```

Per avviare la funzione di simulazione si deve dichiarare la funzione all'interno di OnResetExec per evitare problemi durante la simulazione. Quando viene premuto il pulsante di reset la funzione LCR viene inizializzata e quando viene cliccato il pulsante run il codice di simulazione viene eseguito.

4.3 Configurazione dell'ADC

```
void MX_ADC1_Init(void)
{
  ...
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T1_CC2;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 3;

  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = ADC_REGULAR_RANK_1;

  sConfig.Channel = ADC_CHANNEL_1;
  sConfig.Rank = ADC_REGULAR_RANK_2;

  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = ADC_REGULAR_RANK_3;

  HAL_ADC_Start_DMA(&hadc1, &ADCdata, 3);
}
```

L'assegnazione del valore `ADC_EXTERNALTRIGCONV_T1_CC2` rappresenta che il trigger dell'ADC è impostato sul canale 2 del timer 1. Vengono eseguite 3 conversioni pari alla tensione in uscita, alla corrente nell'induttore e alla tensione del riferimento. Per l'acquisizione della corrente viene controllata la caduta di tensione su un resistore di valore 1 ohm quindi la corrente passante per l'induttore sarà pari alla tensione acquisita dall'ADC. Nella variabile `sConfig.Rank` viene impostato l'ordine con cui vengono acquisiti i vari canali ed è utile per capire in che ordine i dati verranno salvati in memoria.

4.4 Configurazione del timer

```
void MX_TIM1_Init(void)
{
  ...

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_CENTERALIGNED1;
  htim1.Init.Period = 1440;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

   ...

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 720;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_ENABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;

  ...
  sConfigOC.Pulse = 1;

  ...
```

```
  HAL_TIM_MspPostInit(&htim1);
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
        HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_2);
}
```

Il timer andrà alla stessa velocità del clock del µprocessore pari a 72Mhz, non c'è modo di aumentare la velocità, in quanto non sono presenti moltiplicatori di clock. Il primo canale sarà responsabile della generazione del segnale PWM mentre il secondo dell'invio dell'interrupt per l'aquisizione. L'assegnazione del valore `TIM_OCMODE_PWM1` significa che il segnale si trova nella modalità PWM 1 quindi il segnale PWM sarà alto se il contatore è minore del registro di compare e viceversa sarà basso quando è maggiore del valore del registro. Il valore di 1 assegnato alla variabile sConfigOC.Pulse sta a significare che avverrà un interrupt quando il valore del timer sarà pari a 1 che farà campionare l'ADC ed eseguire il codice annesso. Il valore `TIM_COUNTERMODE_CENTERALIGNED1` sta a significare che il contatore conta su e giù alternativamente e che il valore del comparatore verrà confrontato quando il contatore sta contando verso il basso.

4.5 Codice µControllore

```
IL = (ADCdata[0]&0x0000FFFF);         //corrente x1
VC = (ADCdata[0] >> 16);                  //tensione x2
xRef = (ADCdata[1]&0x0000FFFF);  //riferimento tensione
```

Il DMA è resposabile per lo scambio di dati tra ADC e memoria del ucontrollore. Il DMA può scrivere solamente in array di dimensione pari a 32 bit, ma consente di salvare i dati in una lunghezza pari ad una halfword. Utilizzare le halfword consente una maggiore velocità in scrittura rispetto ad usare una word intera. Essendo la tensione dell'ADC espressa in 12 bit si andrà a salvare il dato in 16 bit. Il DMA scriverà nella prima posizione indicata dall'array, salvando il primo dato nei 16 bit meno significativi e il secondo dato nei 16 bit più significativi. Poi passerà alla prossima posizione dell'array e continuerà questo processo, ripetendo la scrittura con lo stesso schema. Per la lettura ci sarà bisogno di recuperare il dato. Si va a salvare i dati di questo array in delle variabili per facilitarne l'utilizzo nel codice; si utilizza una maschera che elimina in 16bit più significativo per prendere il dato nella parte bassa e uno shift a destra di 16 bit per prendere il dato nella parte alta.

```
/*
---- Variabile x2 (tensione VC) --------------------------------
```

```
*/
error2 = xRef - VC;           /* errore di tensione/
/*
I regolatore PI ------------------------------------------------
*/
yp2        = (kp2*error2)>>16;                    /* parte proporzionale*/
yint2 += ki2*error2;                            /* parte integrale (32 bit)*/
x1_ref = yp2 + (yint2>>16);                     /*nuovo riferimento di corrente*/
/*
---- Variabile x1 (Corrente IL) ---------------------------------
*/
//x1_ref = xRef;  //riferimento alla corrente di regime
error1 = x1_ref - IL;      /* errore di corrente1*/
/*
II regolatore PI -----------------------------------------------
*/
yp1         = (kp1*error1)>>16;                 /* parte proporzionale */
yint1 += ki1*error1;                            /*parte integrale (32 bit)*/
/*
-------------------------------------------------------------------
*/
DutyCycle = yp1 + (yint1>>16);

/*
Controllo saturazione  ---------------------------------------
*/
if (DutyCycle <= limL)                  /*controllo limite superiore*/
{
    DutyCycle = limL;
    yint1 = 0;
                }
if (DutyCycle >= limH)                  /*controllo limite inferiore*/
{
    DutyCycle = limH;
    yint1 = 0;
                }
```

```
TIM1->CCR1 = DutyCycle;
```

Nel regolatore PI discreto:

- si calcola l'errore rispetto al riferimento sottraendo il rifermento di tensione dalla tensione in uscita, questo rappresenta l'errore tra il valore voluto e quello reale
- si trova la componente proporzionale moltiplicando l'errore per il guadagno proporzionale poi si fa uno shift a destra di 16 bit per adattare il risultato
- si trova la componente integrale sommando la parte integrale precedentemente calcolata con la moltiplicazione tra guadagno integrale ed errore e si va sempre ad adattare il risultato
- sommando le due parti si ottiene il nuovo riferimento

Con il riferimento di corrente calcolato si fanno gli stessi procedimenti per ottenere il valore del duty cycle. In caso di saturazione del duty cycle si annulla l'integrale del valore della parte integrale del controllo in corrente per evitare un accumulo eccessivo di errore, in modo che il sistema riprenda rapidamente il controllo una volta che il duty cycle rientra nei valori impostati.

La riga di codice `x1_ref = xRef` è utile per bypassare l'anello di tensione per eseguire una taratura del solo anello di corrente, il valore xRef dovrà essere calcolato ed è pari al riferimento alla corrente in uscita per il carico e la tensione della specifica. Il µcontrollore eseguirà comunque il primo controllore PI di tensione, ma il suo risultato calcolato verrà sovrascritto dal riferimento di corrente impostato che è pari a:

$$Iref = \frac{V_{OUT}}{R} = \frac{70V}{2.5\Omega} = 28A$$

Per eseguire una scrittura più sicura ai registri delle periferiche si usano le funzionane di libreria della periferica; queste eseguono una verifica che il registro non sia utilizzato e in uno stato sicuro per la scrittura per poi andare a scrivere nel registro. Per evitare rallentamenti causati dalle librerie si scrive il valore del duty cycle direttamente nel registro CCR1 del timer. Per trovare il registro corretto con facilità si fa una analisi della libreria.

Capitolo 5

Simulazione

5.1 Risposta catena aperta

Per la risposta in catena aperta si imposta il duty cycle a un valore costante. Il duty cycle per avere 70V in uscita con 50V in ingresso è pari a

$$D = 1 - \frac{V_{in}}{V_{out}} = 1 - \frac{50V}{70V} = 0.2857$$

Ora bisogna trovare il valore del duty cycle da salvare nel registro di comparazione CCR1 del timer per avere in uscita una onda quadra con il duty cycle calcolato.

$$CCR1 = periodotimer \cdot dutycycle = 1440 \cdot 0.285 = 411$$

Il periodo timer è pari a quanti passi conta il timer prima tornare a zero e ricominciare a contare da capo. Il valore viene del duty viene scritto direttamente nel registro del timer responsabile della generazione dell'onda PWM con l'istruzione TIM1->CCR1 = 411;

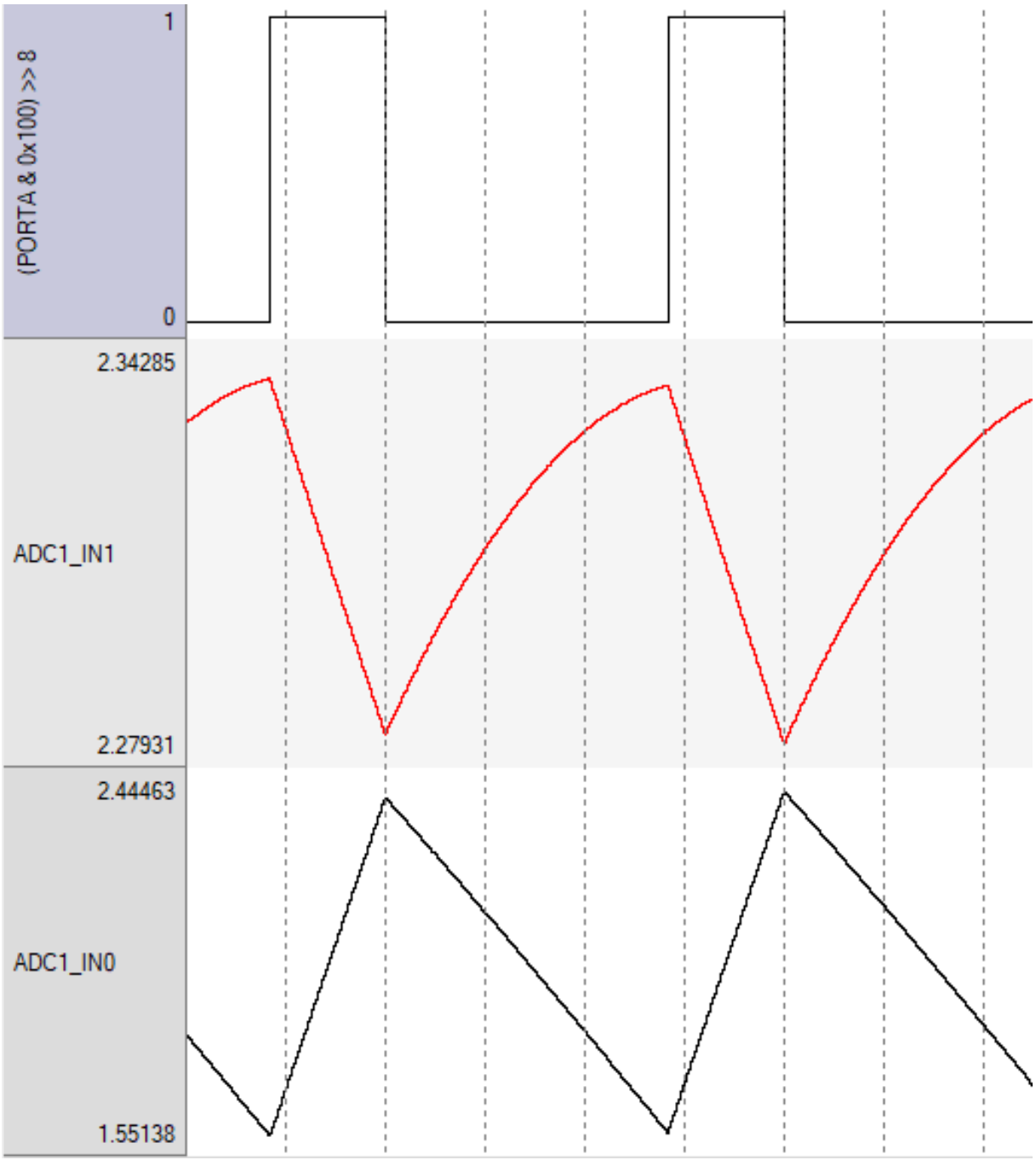


Figura 5.1: risposta catena aperta

La tensione media in uscita è pari a $V_{out} = \frac{100*2.29}{3.3} = 69.39V$, quindi è presente comunque un piccolo errore causato della simulazione che produce solo 240 punti per periodo di commutazione e dalla frequenza del timer, un timer ad una più alta risoluzione consente una precisione di controllo maggiore. Il ripple di tensione è pari a 2.1 V e quello di corrente è uguale a 17.53 A

5.2 Taratura dei guadagni

Il software µvision mette a disposizione un comodo strumento per cambiare i valori delle variabili mentre la simulazione è in esecuzione. Lo strumento in questione è la Watch Windows, dove è possibile vedere il valore delle variabili del programma e cambiarne il valore mentre la simulazione è in esecuzione. Essendo che il segnale di riferimento a gradino e periodico si può notare quasi subito l'effetto del cambio del guadagno consentendo una taratura granulare in poco tempo. Per vedere i segnali simulati si utilizza il Logic Analyzer che consente di avere una rappresentazione grafica nel tempo delle variabili. La memoria dei dati registrati dal Logic Analyser è implementata come un buffer ad anello e ha delle dimensioni limitate pari a 256000 campioni [7]. Non appena tutti i 256k campioni di dati sono stati catturati e visualizzati nel Logic Analyzer i vecchi campioni di dati memorizzati nel buffer di registrazione verranno eliminati. Questo è un grande svantaggio perché se vengono visualizzati i 2 segnali I_L e V_C che vengono simulati con una frequenza di $12Mhz$ si avrà una piccola finestra di visualizzazione pari a $\frac{256000}{12Mhz} \cdot \frac{1}{2} = 10.67ms$. Questo tempo è abbastanza per osservare la risposta al gradino ma se si vuole visualizzare più di 2 segnali si rischia di avere una area di visualizzazione troppo piccola.

Si fa una taratura di ogni blocco singolarmente. Prima si bypassa il controllo di tensione e si applica in ingresso al controllo di corrente direttamente il riferimento della corrente che si è calcolato avere in uscita. Per trovare i vari guadagni si fa una taratura sperimentale della risposta in corrente. Si inizia ponendo il guadagno integrale a 0 e si aumenta il guadagno proporzionale finché non si ottiene una risposta al gradino con una piccola sovra elongazione. Ora si avrà una risposta al gradino con un grande errore e si deve andare ad alzare il guadagno integrale finché si ottiene un errore nullo, in poco tempo, con una risposta del primo ordine o del secondo ordine ma molto smorzata. Poi si va a modificare il guadagno proporzionale per rendere la risposta più smorzata.

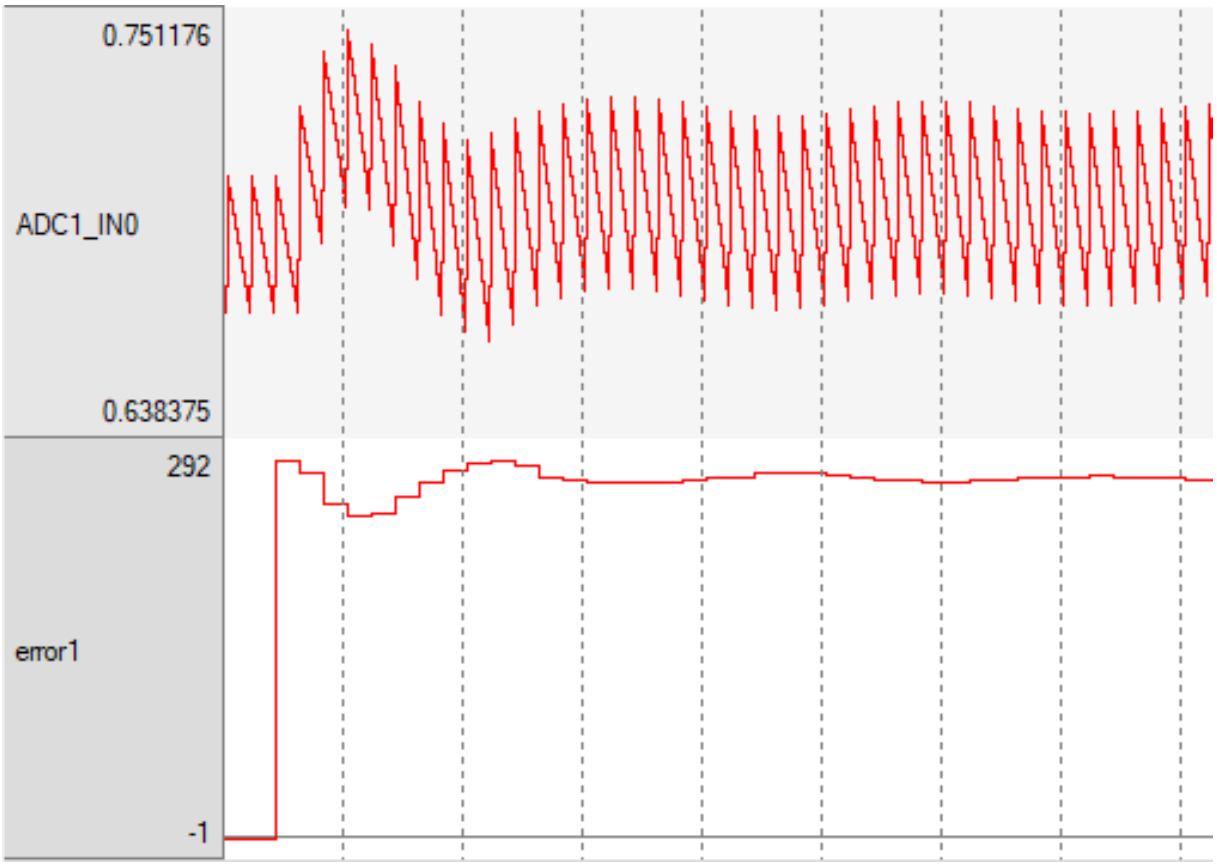


Figura 5.2: Risposta con Kp1=10000 e controllo in tensione superato

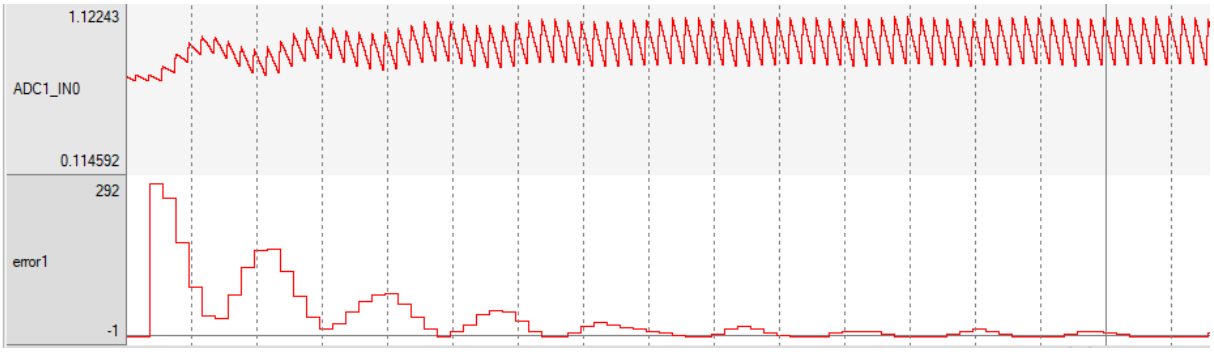


Figura 5.3: Risposta con Kp1=10000 e Ki1=3000 e controllo in tensione superato

La risposta in corrente presenta molte oscillazioni ma sono inevitabili per avere una risposta veloce, sarà compito del controllo in tensione di ammortizzare le oscillazioni.

Si toglie il bypass e si fa la taratura del controllo di tensione con i guadagni del controllo di corrente già trovati precedentemente

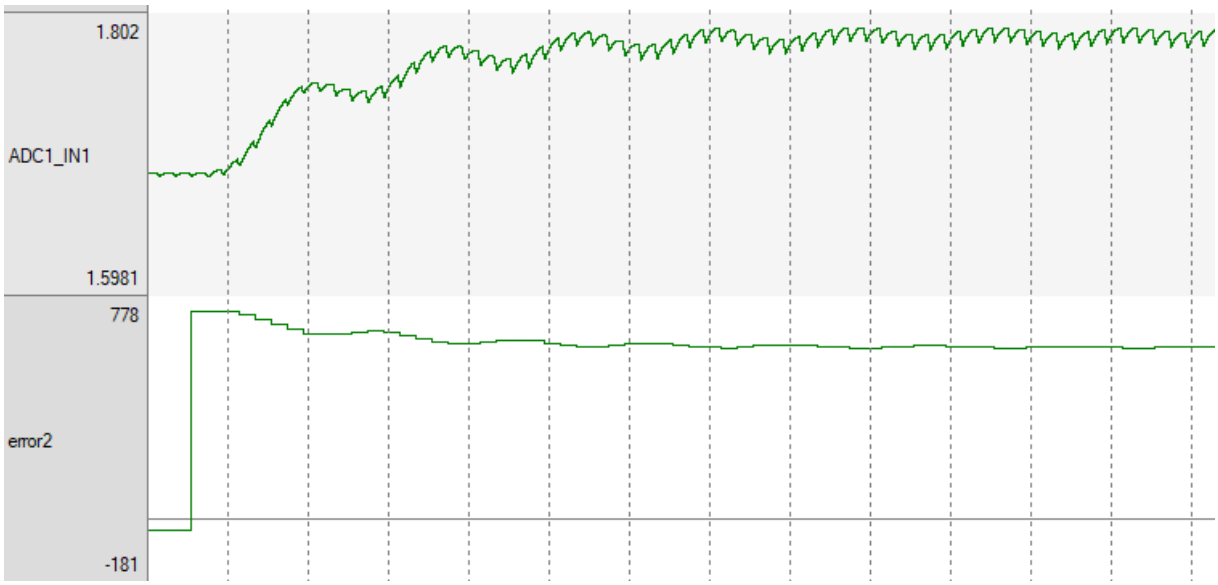


Figura 5.4: Kp2=35000

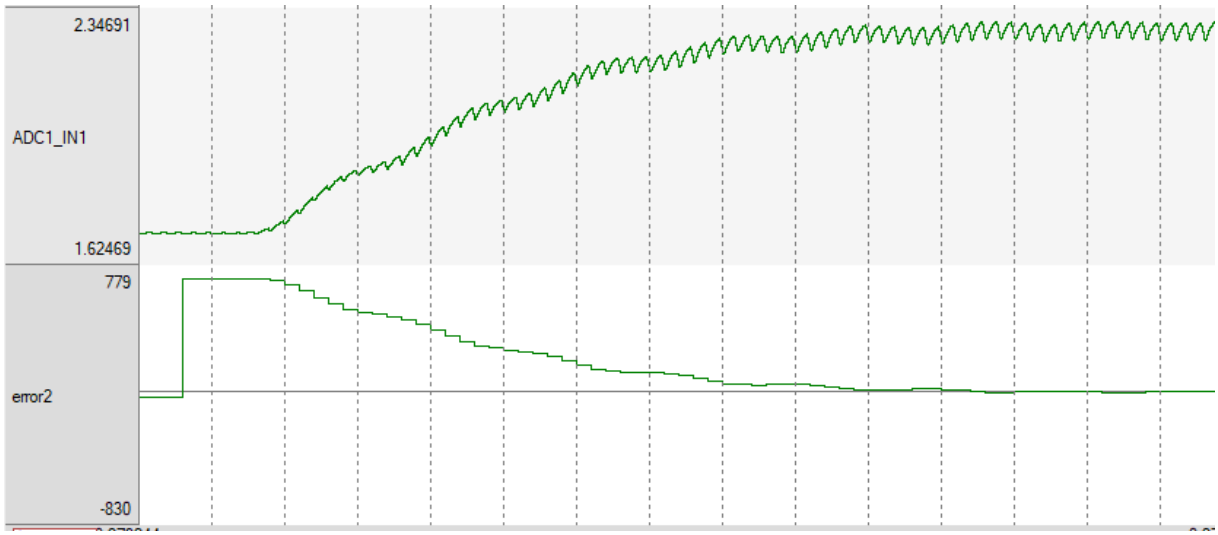


Figura 5.5: Kp2=35000 ki2=2500

A regime l'errore è quasi nullo e la tensione media è pari a 69.81 V. Dopo l'applicazione di un gradino, il tempo di assestamento è di 2.2 millisecondi, durante il quale la tensione raggiunge l'1% del valore finale, e l'overshoot è praticamente nullo.

5.3 Risposta catena chiusa

5.4 Risposta al cambio di carico

Per la risposta al cambio di carico si analizza il caso in cui il carico diminuisca da un valore di 2.5Ω ad un valore di 1.25Ω. Questo perché è la casistica che avrà l'overshoot maggiore causata dalle grandi differenze di corrente in gioco.

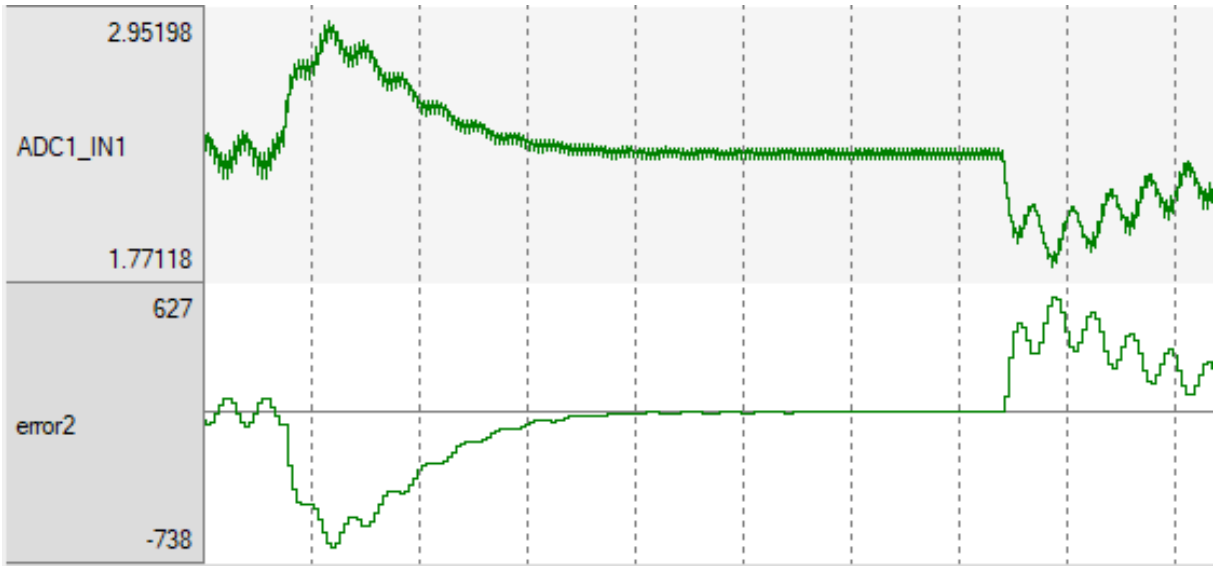


Figura 5.6: Risposta con C=182µF

Senza modificare i guadagni trovati precedentemente si ottiene una risposta del secondo ordine molto oscillante. Questo comportamento è causato dal valore elevato dell'induttanza che ha molta energia immagazzinata che tende a mantenere constante la corrente che le scorre attraverso. Si aumenta la capacità del condensatore e di conseguenza si diminuisce il valore dell'induttore per tenere la frequenza di taglio allo stesso valore secondo la formula

$$L = \frac{1}{\omega^2 C} = \frac{1}{(2\pi \cdot 2146)^2 \cdot 300 \cdot 10^{-6}} = 20.7\mu H$$

Questo provocherà un rallentamento della risposta del sistema al gradino ma una migliore risposta al cambio di carico causata dalla riduzione di energia immagazzinata nell'induttore.

Una diminuzione del valore dell'induttanza provoca però una aumento del ripple di corrente nell'induttanza. Si avrà poco ripple nella corrente in uscita perché verrà in parte ammortizzato dal filtro LC.

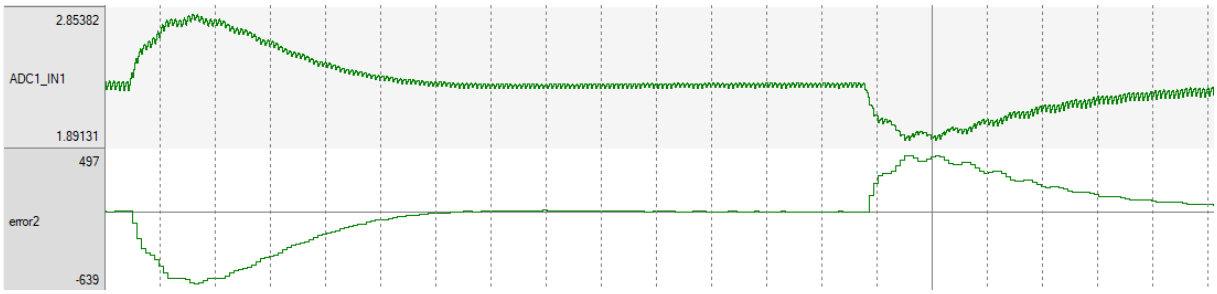


Figura 5.7: Risposta con C=300µF

Dopo il cambio di carico, il tempo di assestamento è di 3.3 millisecondi, durante il quale la tensione raggiunge l'1% del valore finale e sono presenti meno oscillazioni rispetto ad avere un condensatore più piccolo. L'overshoot è presente e la tensione raggiunge un valore minimo di 57.27 Volt quando il carico aumenta e un valore massimo di 86.36 V quando il carico diminuisce.

Capitolo 6

Conclusioni

Lo studio di questo convertitore si è concentrato soprattutto sull'utilizzo del software di simulazione fornito dal programma µVision. Si è vista la facilità di creare un modello software per fare dei controlli iniziali prima di implementare il progetto sull'hardware riducendo i costi causati da modifiche del circuito o rischi causati da malfunzionamenti inaspettati. Per migliorare il circuito si può modificare l'interruttore elettronico, essendo le correnti molto alte e la frequenza di switch bassa un interruttore digitale di tipo IGBT può essere preferibile rispetto a un MOSFET. Un esempio di una applicazione pratica di questo progetto può essere il controllo di un motore di un nastro trasportatore con carico variabile.

Bibliografia

[1] «Basic Calculation of a Boost Converter's Power Stage.» (), indirizzo: https://www.ti.com/lit/an/slva372d/slva372d.pdf?ts=1721454949874.

[2] «Data sheet STM32F103x8.» (), indirizzo: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

[3] «Reference manual.» (), indirizzo: https://www.st.com/resource/en/reference_manual/rm0008-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf.

[4] «Debug Functions.» (), indirizzo: https://developer.arm.com/documentation/101407/0540/Debug-Functions.

[5] «Legacy Device List.» (), indirizzo: https://www.keil.com/dd/.

[6] «UVISION DEBUGGER: Simulation of Cortex-M Devices.» (), indirizzo: https://developer.arm.com/documentation/ka002225/latest/.

[7] «UVISION DEBUGGER: Sample buffer size of Logic Analyzer in MDK uVision.» (), indirizzo: https://developer.arm.com/documentation/ka004095/latest/.